Problem Set 6: Predicate Transforms

Hints and Notes

1. Consider the assertion of *weak* correctness: $\{t=5 \wedge z<0\}$ s $\{y=z+1 \wedge t=z\}$. Which of the following observations/facts would allow one to deduce that the assertion is FALSE and which would not?

   a. The wp(s, y=z) is $z>-5$.

   b. The wlp(s, y=z) is $z>-5$.

1. Consider the assertion of *weak* correctness: $\{t=5 \wedge z<0\}$ s $\{y=z+1 \wedge t=z\}$. Which of the following observations/facts would allow one to deduce that the assertion is FALSE and which would not?

a. The wp(s, y=z) is z>-5.

b. The wlp(s, y=z) is z>-5.

1. Consider the assertion of *weak* correctness: {t=5 Λ z<0} s {y=z+1 Λ t=z}. Which of the following observations/facts would allow one to deduce that the assertion is FALSE and which would not?

a. The wp(s, y=z) is z>-5.

Would. [wp(s, **y=z**) = z>-5] => [{t=5 Λ -5<z<0} s {**y=z**} *strongly*] => [{t=5 Λ z<0} s {**y=z+1** Λ t=z} is false] since we know s will halt for at least four initial (integer) values of z<0 with post condition (**y=z+1** Λ t=z) being false.

b. The wlp(s, y=z) is z>-5.

1. Consider the assertion of *weak* correctness: $\{t=5 \wedge z<0\}$ s $\{y=z+1 \wedge t=z\}$. Which of the following observations/facts would allow one to deduce that the assertion is FALSE and which would not?

   a. The wp(s, y=z) is z>-5.

      Would. [wp(s, **y=z**) = z>-5] => [$\{t=5 \wedge -5<z<0\}$ s {**y=z**} *strongly*] => [$\{t=5 \wedge z<0\}$ s {**y=z+1** $\wedge$ t=z} is false] since we know s will halt for at least four initial (integer) values of z<0 with post condition (**y=z+1** $\wedge$ t=z) being false.

   b. The wlp(s, y=z) is z>-5.

1. Consider the assertion of *weak* correctness: $\{t=5 \land z<0\}$ s $\{y=z+1 \land t=z\}$. Which of the following observations/facts would allow one to deduce that the assertion is FALSE and which would not?

   a. The wp(s, y=z) is z>-5.

      Would. [wp(s, **y=z**) = z>-5] => [{t=5 ∧ -5<z<0} s {**y=z**} *strongly*] => [{t=5 ∧ z<0} s {**y=z+1** ∧ t=z} is false] since we know s will halt for at least four initial (integer) values of z<0 with post condition (**y=z+1** ∧ t=z) being false.

   b. The wlp(s, y=z) is z>-5.

      Would not. [wlp(s, **y=z**) = z>-5] => [{t=5 ∧ z>-5} s {**y=z**}] *weakly*. But this does **not** imply that {t=5 ∧ z<0} s {**y=z+1** ∧ t=z} is FALSE since s may not terminate for -5<z<0.

2. Consider the program, S, below:

```
temp := x
x := y
y := temp
if temp>z then
    y := z
    z := temp
    if x>y then
        temp := x
        x := y
        y := temp
    end_if
end_if
```

Under what circumstances will the program result in $\{x \le y < z\}$? (Hint: determine the wp of the program w.r.t. the desired result.)

2. Consider the program, S, below:

```
          temp := x
S1        x := y
          y := temp
          if temp>z then
              y := z
              z := temp
              if x>y then
S2                temp := x
                  x := y
                  y := temp
              end_if
          end_if
```

Under what circumstances will the program result in {x≤y<z}? (Hint: determine the wp of the program w.r.t. the desired result.)

2. Consider the program, S, below:

```
                temp := x
S1              x := y
                y := temp
                if temp>z then
        S2,1      y := z
                  z := temp
                  if x>y then
S2                    temp := x
        S2,2          x := y
                      y := temp
                  end_if
                end_if
```

Under what circumstances will the program result in $\{x \le y < z\}$? (Hint: determine the wp of the program w.r.t. the desired result.)

2. Consider the program, S, below:

```
S1      temp := x
        x := y
        y := temp
S2      if temp>z then
  S2,1      y := z
            z := temp
  S2,2      if x>y then
                temp := x
                x := y        S2,2,1
                y := temp
            end_if
        end_if
```

Under what circumstances will the program result in {x≤y<z}? (Hint: determine the wp of the program w.r.t. the desired result.)

2. Consider the program, S, below:

```
S1      temp := x
        x := y
        y := temp
S2      if temp>z then
S2,1        y := z
            z := temp
S2,2        if x>y then
                temp := x
                x := y          S2,2,1
                y := temp
            end_if
        end_if
```

$$wp(S, x \le y < z) = wp(S1;\ S2, x \le y < z)$$

2. Consider the program, S, below:

```
S1        temp := x
          x := y
          y := temp
S2        if temp>z then
  S2,1       y := z
             z := temp
  S2,2       if x>y then
                temp := x
                x := y          S2,2,1
                y := temp
             end_if
          end_if
```

$$wp(S, x \le y < z) = wp(S1;\, S2, x \le y < z)$$

$$= wp(S1, wp(S2, x \le y < z))$$

2. Consider the program, S, below:

```
           temp := x
S1         x := y
           y := temp
           if temp>z then
      S2,1    y := z
              z := temp
              if x>y then
S2                temp := x
      S2,2        x := y          S2,2,1
                  y := temp
              end_if
           end_if
```

$$wp(S, x \le y < z) = wp(S1;\ S2, x \le y < z)$$

$$= wp(S1, wp(S2, x \le y < z))$$

$$= wp(S1, wp(\text{if } temp > z \text{ then } S2.1; S2.2,\ x \le y < z))$$

2. Consider the program, S, below:

```
                temp := x
S1              x := y
                y := temp
                if temp>z then
        S2,1      y := z
                  z := temp
S2                if x>y then
                     temp := x
        S2,2         x := y          S2,2,1
                     y := temp
                  end_if
                end_if
```

$$
\begin{aligned}
wp(S, x \le y < z) &= wp(S1;\ S2,\ x \le y < z) \\
&= wp(S1,\ wp(S2,\ x \le y < z)) \\
&= wp(S1,\ wp(\text{if } temp > z \text{ then } S2.1; S2.2, \\
&\qquad\qquad x \le y < z))
\end{aligned}
$$

etc.

3. We have learned that P => wp(s,Q) implies {P} s {Q}. However, {P} s {Q} does NOT imply P => wp(s,Q). Why?

3. We have learned that P => wp(s,Q) implies {P} s {Q}. However, {P} s {Q} does NOT imply P => wp(s,Q). Why?

Because:

P => wp(s,Q) <=> "{P} s {Q} *strongly*"

3. We have learned that P => wp(s,Q) implies {P} s {Q}. However, {P} s {Q} does NOT imply P => wp(s,Q). Why?

Because:

P => wp(s,Q) <=> "{P} s {Q} *strongly*"

and

"{P} s {Q} *strongly*" => {P} s {Q}

3. We have learned that P => wp(s,Q) implies {P} s {Q}. However, {P} s {Q} does NOT imply P => wp(s,Q). Why?

Because:

P => wp(s,Q) <=> "{P} s {Q} *strongly*"

and

"{P} s {Q} *strongly*" => {P} s {Q}

but

{P} s {Q} ≠> "{P} s {Q} *strongly*".

4. a. Use the wlp rule for while-do statements given in Lecture Notes 20 to find the weakest liberal pre-condition of the following program with respect to the post-condition Z=XY.

```
while J<>Y do
   Z := Z+X;
   J := J+1
end_while
```

b. Consider the invariant, I: Z=XJ, used with the while-loop ROI to prove the assertion given on slide 11 of Lecture Notes 18. How does it compare in "strength“ to the weakest liberal pre-condition from part (a) above? (In particular, is one STRONGER than the other?) Briefly explain your answer.

a. wlp rule for while-do statements:

$$\textbf{wlp(while b do S, Q)} \equiv \textbf{wp(while b do S, Q)} \lor \neg\textbf{wp(while b do S, true)}$$

a. wlp rule for while-do statements:

wlp(while b do S, Q) ≡ wp(while b do S, Q) V ¬wp(while b do S, true)

i. determining wp(while b do S, Q):

```
while J<>Y do
   Z := Z+X;
   J := J+1
end_while
```

H_0: $J=Y \wedge Z=XY$

a. wlp rule for while-do statements:

wlp(while b do S, Q) ≡ wp(while b do S, Q) V ¬wp(while b do S, true)

i. determining wp(while b do S, Q):

```
while J<>Y do
   Z := Z+X;
   J := J+1
end_while
```

H_0: $J=Y \wedge Z=XY$

H_1: $J<>Y \wedge wp(s, J=Y \wedge Z=XY)$

$= J=Y-1 \wedge Z=X(Y-1)$

a. wlp rule for while-do statements:

wlp(while b do S, Q) ≡ wp(while b do S, Q) V ¬wp(while b do S, true)

i. determining wp(while b do S, Q):

H_0: J=Y ∧ Z=XY

H_1: J<>Y ∧ wp(s, J=Y ∧ Z=XY)
= J=Y-1 ∧ Z=X(Y-1)

H_2: J<>y ∧ wp(s, J=Y-1 ∧ Z=X(Y-1))
= J=Y-2 ∧ Z=X(Y-2)

```
while J<>Y do
   Z := Z+X;
   J := J+1
end_while
```

a. wlp rule for while-do statements:

wlp(while b do S, Q) ≡ wp(while b do S, Q) ∨ ¬wp(while b do S, true)

i. determining wp(while b do S, Q):

```
while J<>Y do
   Z := Z+X;
   J := J+1
end_while
```

H_0: $J=Y \wedge Z=XY$

H_1: $J<>Y \wedge wp(s, J=Y \wedge Z=XY)$
$= J=Y-1 \wedge Z=X(Y-1)$

H_2: $J<>y \wedge wp(s, J=Y-1 \wedge Z=X(Y-1))$
$= J=Y-2 \wedge Z=X(Y-2)$

H_k: $J=Y-k \wedge Z=X(Y-k)$

a. wlp rule for while-do statements:

$$\mathbf{wlp(while\ b\ do\ S,\ Q)\ \equiv\ wp(while\ b\ do\ S,\ Q)\ \vee\ \neg wp(while\ b\ do\ S,\ true)}$$

i. determining wp(while b do S, Q):

```
while J<>Y do
   Z := Z+X;
   J := J+1
end_while
```

H_0: $J=Y \wedge Z=XY$

H_1: $J<>Y \wedge wp(s, J=Y \wedge Z=XY)$
$= J=Y-1 \wedge Z=X(Y-1)$

H_2: $J<>y \wedge wp(s, J=Y-1 \wedge Z=X(Y-1))$
$= J=Y-2 \wedge Z=X(Y-2)$

H_k: $J=Y-k \wedge Z=X(Y-k)$

Therefore, $H_0 \vee H_1 \vee H_2 \vee \ldots \vee H_k \vee \ldots$ simplifies to:

$$(J=Y \wedge Z=XY) \vee (J<Y \wedge Z=XJ) \;=\; \mathbf{(J\leq Y \wedge Z=XJ)}$$

ii. determining wp(while b do S, true):

ii. determining wp(while b do S, true):

H_0: $J=Y \land true$

```
while J<>Y do
   Z := Z+X;
   J := J+1
end_while
```

ii. determining wp(while b do S, true):

H_0: J=Y ∧ true

H_1: J<>Y ∧ wp(s, J=Y)

= J=Y-1

```
while J<>Y do
   Z := Z+X;
   J := J+1
end_while
```

ii. determining wp(while b do S, true):

H_0: J=Y ∧ true

H_1: J<>Y ∧ wp(s, J=Y)
= J=Y-1

H_2: J<>y ∧ wp(s, J=Y-1)
= J=Y-2

```
while J<>Y do
   Z := Z+X;
   J := J+1
end_while
```

ii. determining wp(while b do S, true):

H_0: J=Y ∧ true

H_1: J<>Y ∧ wp(s, J=Y)
= J=Y-1

H_2: J<>y ∧ wp(s, J=Y-1)
= J=Y-2

H_k: J=Y-k

```
while J<>Y do
   Z := Z+X;
   J := J+1
end_while
```

ii. determining wp(while b do S, true):

H_0: $J=Y \wedge true$

H_1: $J<>Y \wedge wp(s, J=Y)$
$= J=Y-1$

H_2: $J<>y \wedge wp(s, J=Y-1)$
$= J=Y-2$

H_k: $J=Y-k$

```
while J<>Y do
   Z := Z+X;
   J := J+1
end_while
```

Therefore, $H_0 \vee H_1 \vee H_2 \vee \ldots \vee H_k \vee \ldots$ simplifies to: $\mathbf{J \leq Y}$

ii. determining wp(while b do S, true):

H_0: J=Y ∧ true

H_1: J<>Y ∧ wp(s, J=Y)
= J=Y-1

H_2: J<>y ∧ wp(s, J=Y-1)
= J=Y-2

H_k: J=Y-k

```
while J<>Y do
   Z := Z+X;
   J := J+1
end_while
```

Therefore, H_0 V H_1 V H_2 V ... V H_k V ... simplifies to: **J≤Y**

Thus, wlp(while b do S, Q) = (J≤Y ∧ Z=XJ) V J>Y
= **Z=XJ V J>Y**

6. a. Identify H_1, H_2, H_3, and H_k such that

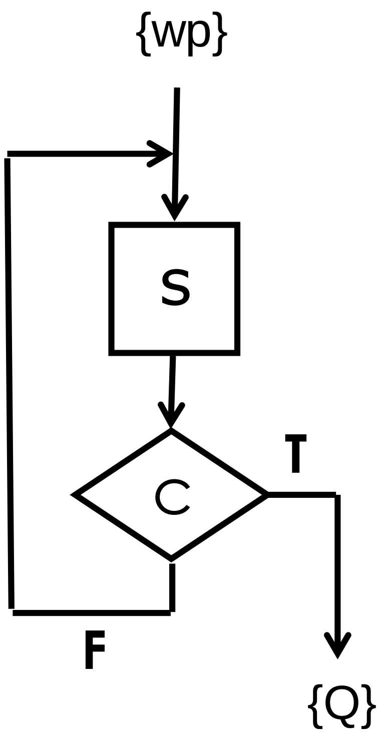
$$wp(\text{Repeat s Until c}, Q) = H_1 \vee H_2 \vee H_3 \vee \ldots \vee H_k \vee \ldots$$

where H_i represents the necessary and sufficient condition that "Repeat s Until c“ terminates in state Q after i executions of s. For i>1, H_i should be expressed as a function of H_{i-1}.

6. a. Identify H_1, H_2, H_3, and H_k such that

$$wp(\text{Repeat s Until c}, Q) = H_1 \vee H_2 \vee H_3 \vee \ldots \vee H_k \vee \ldots$$

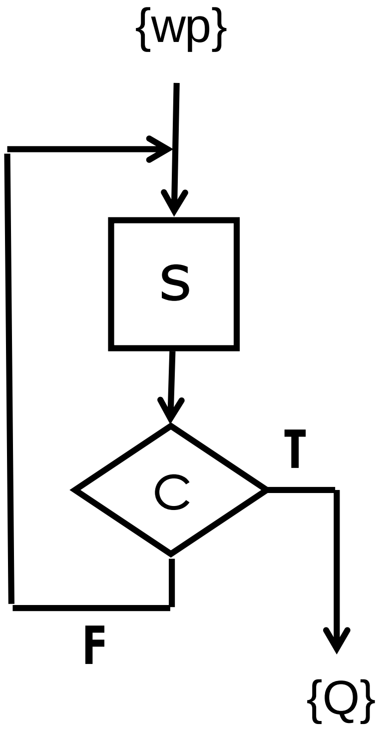
where H_i represents the necessary and sufficient condition that "Repeat s Until c“ terminates in state Q after i executions of s. For i>1, H_i should be expressed as a function of H_{i-1}.



6. a. Identify H_1, H_2, H_3, and H_k such that

$$wp(\text{Repeat s Until c}, Q) = H_1 \vee H_2 \vee H_3 \vee \ldots \vee H_k \vee \ldots$$

where H_i represents the necessary and sufficient condition that "Repeat s Until c" terminates in state Q after i executions of s. For i>1, H_i should be expressed as a function of H_{i-1}.

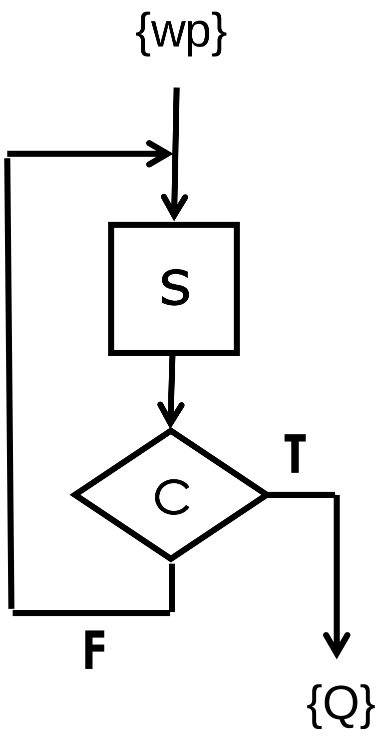


H_1 = ? (s executes once only)

6. a. Identify H_1, H_2, H_3, and H_k such that

$$wp(\text{Repeat s Until c}, Q) = H_1 \vee H_2 \vee H_3 \vee \ldots \vee H_k \vee \ldots$$

where H_i represents the necessary and sufficient condition that "Repeat s Until c“ terminates in state Q after i executions of s. For i>1, H_i should be expressed as a function of H_{i-1}.

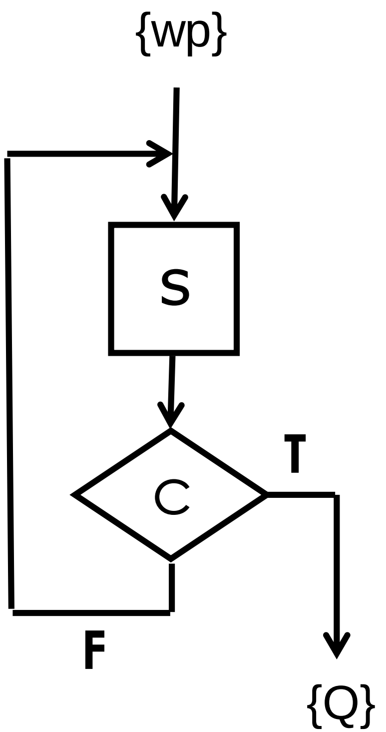


$H_1 = wp(s, ?)$

6. a. Identify H_1, H_2, H_3, and H_k such that

$$wp(\text{Repeat s Until c}, Q) = H_1 \lor H_2 \lor H_3 \lor \ldots \lor H_k \lor \ldots$$

where H_i represents the necessary and sufficient condition that "Repeat s Until c" terminates in state Q after i executions of s. For i>1, H_i should be expressed as a function of H_{i-1}.

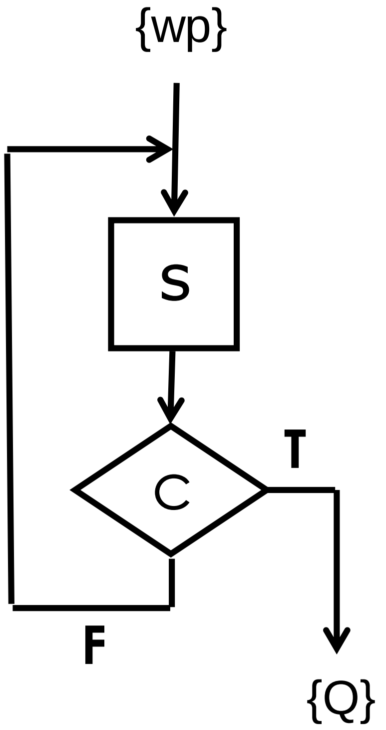


$H_1 = wp(s, c \land Q)$

6. a. Identify H_1, H_2, H_3, and H_k such that

$$wp(\text{Repeat s Until c, Q}) = H_1 \vee H_2 \vee H_3 \vee ... \vee H_k \vee ...$$

where H_i represents the necessary and sufficient condition that "Repeat s Until c" terminates in state Q after i executions of s. For i>1, H_i should be expressed as a function of H_{i-1}.



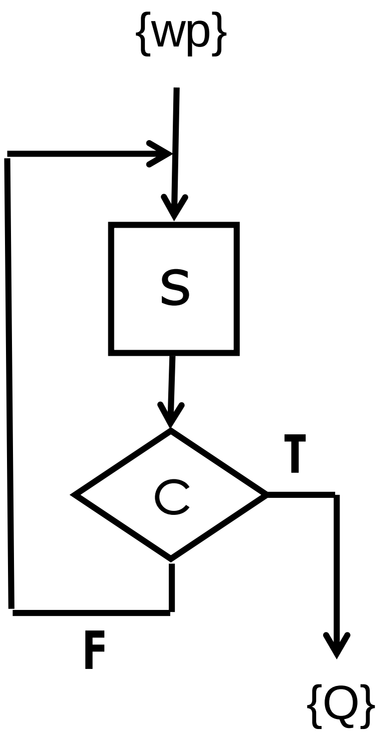
H_1 = wp(s, c $\wedge$ Q)

H_2 = ? (s executes twice only)

6. a. Identify H_1, H_2, H_3, and H_k such that

$$wp(\text{Repeat s Until c, Q}) = H_1 \vee H_2 \vee H_3 \vee \ldots \vee H_k \vee \ldots$$

where H_i represents the necessary and sufficient condition that "Repeat s Until c" terminates in state Q after i executions of s. For i>1, H_i should be expressed as a function of H_{i-1}.



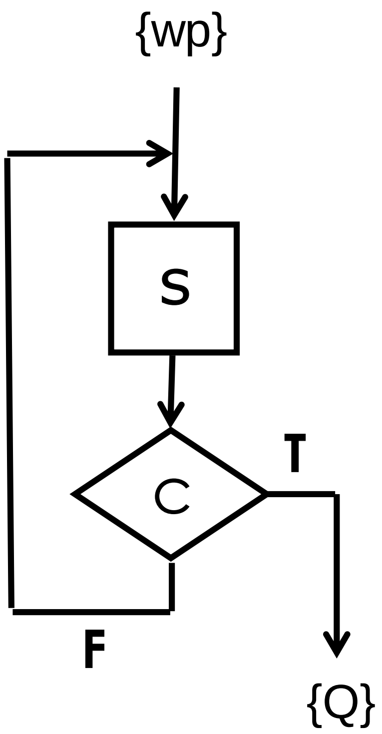
$H_1 = wp(s, c \wedge Q)$

$H_2 = wp(s, \neg c \wedge ?)$

6. a. Identify H_1, H_2, H_3, and H_k such that

$$wp(\text{Repeat s Until c}, Q) = H_1 \vee H_2 \vee H_3 \vee \ldots \vee H_k \vee \ldots$$

where H_i represents the necessary and sufficient condition that "Repeat s Until c" terminates in state Q after i executions of s. For i>1, H_i should be expressed as a function of H_{i-1}.



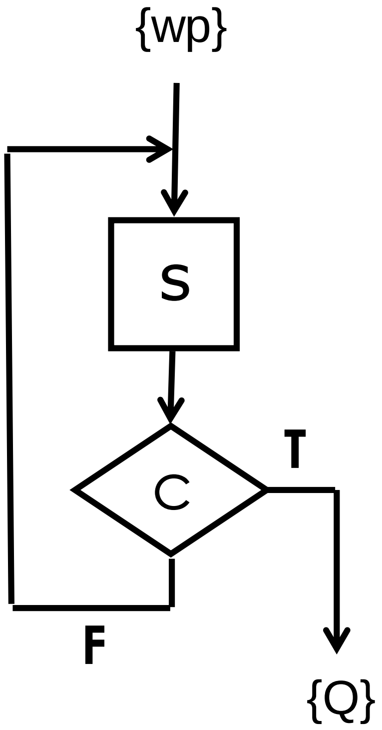
$H_1 = wp(s, c \wedge Q)$

$H_2 = wp(s, \neg c \wedge wp(s, c \wedge Q))$

6. a. Identify H_1, H_2, H_3, and H_k such that

$$wp(\text{Repeat s Until c}, Q) = H_1 \vee H_2 \vee H_3 \vee \ldots \vee H_k \vee \ldots$$

where H_i represents the necessary and sufficient condition that "Repeat s Until c" terminates in state Q after i executions of s. For i>1, H_i should be expressed as a function of H_{i-1}.



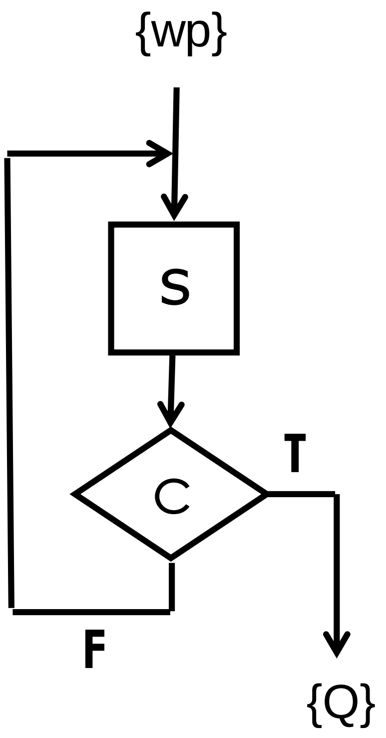
$H_1 = \underline{wp(s, c \wedge Q)}$

$H_2 = wp(s, \neg c \wedge \underline{wp(s, c \wedge Q)})$

6. a. Identify H_1, H_2, H_3, and H_k such that

$$wp(\text{Repeat s Until c}, Q) = H_1 \vee H_2 \vee H_3 \vee \ldots \vee H_k \vee \ldots$$

where H_i represents the necessary and sufficient condition that "Repeat s Until c“ terminates in state Q after i executions of s. For i>1, H_i should be expressed as a function of H_{i-1}.

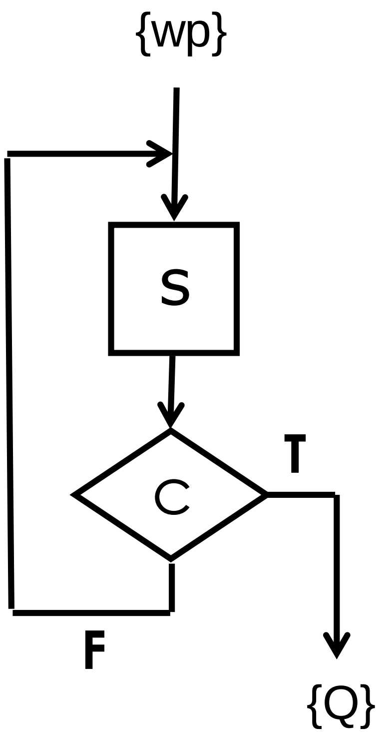


$$H_1 = wp(s, c \wedge Q)$$
$$H_2 = wp(s, \neg c \wedge wp(s, c \wedge Q))$$
$$= wp(s, \neg c \wedge H_1)$$

6. a. Identify H_1, H_2, H_3, and H_k such that

$$wp(\text{Repeat s Until c}, Q) = H_1 \vee H_2 \vee H_3 \vee \ldots \vee H_k \vee \ldots$$

where H_i represents the necessary and sufficient condition that "Repeat s Until c“ terminates in state Q after i executions of s. For i>1, H_i should be expressed as a function of H_{i-1}.

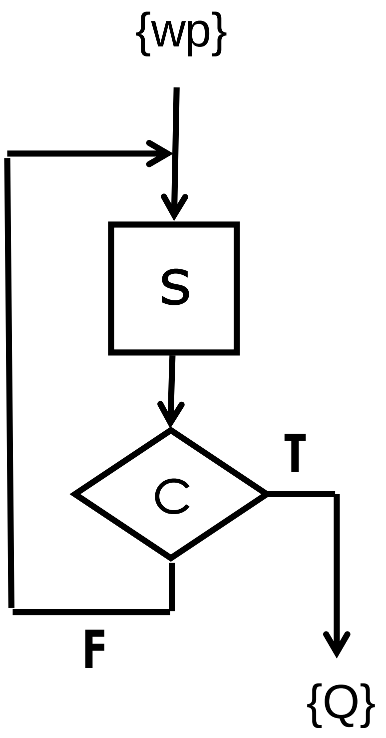


$$H_1 = wp(s, c \wedge Q)$$
$$H_2 = wp(s, \neg c \wedge wp(s, c \wedge Q))$$
$$= wp(s, \neg c \wedge H_1)$$
$$H_3 = wp(s, \neg c \wedge H_2)$$

6. a. Identify H_1, H_2, H_3, and H_k such that

$$wp(\text{Repeat s Until c}, Q) = H_1 \vee H_2 \vee H_3 \vee \ldots \vee H_k \vee \ldots$$

where H_i represents the necessary and sufficient condition that "Repeat s Until c“ terminates in state Q after i executions of s. For i>1, H_i should be expressed as a function of H_{i-1}.



$$H_1 = wp(s, c \wedge Q)$$
$$H_2 = wp(s, \neg c \wedge wp(s, c \wedge Q))$$
$$= wp(s, \neg c \wedge H_1)$$
$$H_3 = wp(s, \neg c \wedge H_2)$$

$$H_k = wp(s, \neg c \wedge H_{k-1})$$

Problem Set 6: Predicate Transforms

Hints and Notes